METHOD AND APPARATUS FOR MANAGING JOB QUEUES

BACKGROUND

In a networked office environment, print jobs are queued on a network server before being passed to a device for printing. New printers and Multifunction Peripheral (MFP) devices also include queue for storing print jobs. Because there are two different queuing points in the same job stream, there is no way for a user or administrator to effectively view and manage the jobs or both queues at the same time.

Currently a user has to use separate programs with separate User Interfaces (UIs) to manage jobs on the network server and on the peripheral device. A network queue on the network server is typically managed using Network Operating System (NOS) software tools such as created by Novell or Microsoft. A device queue on the peripheral device is typically managed using vendor specific application software that uses a proprietary or standardized network protocol such as Simple Network Management Protocol (SNMP)/Job Management Information Base (MIB) or Internet Printing Protocol (IPP).

None of these user interfaces can manage more than one job queue at the same time. Thus, a user cannot effectively manipulate the scheduling of jobs from multiple job queues. The present invention addresses this and other problems associated with the prior art.

SUMMARY OF THE INVENTION

A queue manager monitors a server queue in a network server and a device queue in a peripheral device at the same time. A user interface displays the status of jobs in the server

queue and device queue on the same display and allows a user to manipulate any of the jobs on either queue using the same user interface.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of a queue management system.

FIG. 2 is a diagram showing a user interface for the queue management system shown in FIG. 1.

FIG. 3 is a diagram showing an alternative user interface for the queue management system shown in FIG. 1.

FIG. 4 is a flow diagram showing how jobs are cancelled or paused by a queue manager.

FIG. 5 are diagrams showing in further detail how the queue manager cancels or pauses jobs.

FIG. 6 is a flow diagram showing how the priority of jobs are changed by the queue manager.

FIG. 7 is a diagram showing in further detail how the priority of jobs are changed by the queue manager.

DETAILED DESCRIPTION

A queue management system 12 includes a computer 14 that is connected through a network 18 to a network server 22 and a peripheral device 26. The computer 14 can be any Personal Computer (PC), terminal, etc. or can be part of the network server 22 or peripheral device 26. The computer 14 includes a processor 13 that accesses a queue manager program 15 that is stored in computer memory. The processor 13 uses a User Interface (UI) 16 to

receive user inputs and to display status for jobs stored on network sever 22 and peripheral device 26.

The network server 22 can be any computing device that is used by network 18 to store jobs in a server queue 24. The peripheral device 26 is any copier, printer, fax machine, scanner, MultiFunction Peripheral (MFP) device, or other output device that can output copy, print, fax, and/or scan jobs. The peripheral device 26 includes a device queue 28. The network 18 can be any Internet or Intranet network such as a Wide Area Network (WAN) Internet Protocol (IP) network or a Local Area Network (LAN) Ethernet network.

A job 20 is sent from computer 14 over network 18 to the server queue 24. The job 20 in this example is a print job but can be any copy, print, fax, scan, etc. job that needs to be output by peripheral device 26. The print job 20 is eventually moved from server queue 24 to the device queue 28. The device queue 28 may hold many different jobs that may come from many different sources other than computer 14.

The user interface 16 operated by the queue manager 15 provides information on the jobs 17 held in each of the two queues 24 and 28. This allows a user or administrator to view the order and status of the combined set of jobs without concern for which queue they reside in. The queue manager 15 allows a user or administrator to manage jobs 17 in the two queues seamlessly as if they were in a single queue. The queue manager 15 allows a user to delete, pause, hold or reorder the priority of jobs in both queue 24 and queue 28 through the same user interface 16. The queue manager 15 can manage any number of job queues that may exist on the network 18.

FIG. 2 is a more detailed drawing of the user interface 16. In one embodiment, a document name field 30 lists the different names of the jobs DOC1 - DOC6 in order of output priority. A status field 32 identifies the status of the jobs such as printing, waiting, holding,

etc. A location field 34 identifies the device where the job is currently residing. For example, job DOC 1 is currently residing in a MFP peripheral device queue and job DOC 4 is currently residing in a network server queue. Owner field 36 identifies a user that initiated the job. Other fields 38 associated with the jobs may also be displayed on the user interface 16.

A user can change the priority of the jobs simply by dragging the jobs to different locations on user interface 16. For example, a user may move a cursor over job DOC5 and click a mouse button. The user can then drag the cursor to a location between job DOC2 and DOC3 and release the mouse button. The queue manager 15 performs the necessary commands to move job DOC5 up in priority above jobs DOC3 and DOC4.

To cancel a job, the user selects the job on user interface 16 and then enters a delete command. For example, the user moves the cursor over job DOC 3 and initiates a delete command from a computer menu. The queue manager 15 then sends the necessary commands to the MFP queue to cancel job DOC3.

FIG. 3 shows an alternate implementation of user interface 16 where the jobs are displayed according to the device where the job is currently residing. In this example, jobs DOC1, DOC2, and DOC 3 all currently reside in the device queue 28 in peripheral device 26 (FIG. 1). The DOC1 job has the highest priority in device queue 28 and is currently printing. Jobs DOC4, DOC5 and DOC 6 are currently residing in the server queue 24 in network server 22 (FIG. 1).

FIG. 4 is a flow diagram showing how jobs are cancelled or paused by the queue manager 15. The queue manager displays the status of jobs in the different queues in block 40. In this example, the device queue 28 and the server queue 24 are both sent inquiries 52 from the queue manager 15. The identity of jobs in device queue 28 and server queue 24 are

sent back to the queue manager 15 in replies 54. The queue manager 15 then uses the information in replies 54 to update the job list output on the user interface.

A user may select one or more of the displayed jobs to cancel or pause in block 42. The queue manager sends out a cancel or pause request to one or both of the device queue 28 and server queue 24 in block 44. The queue manager waits in block 46 for a confirmation from the queue currently storing the job confirming that the selected job has been cancelled or paused. When the confirmation is received, the queue manager updates the job list on the user interface display in block 48 to remove the cancelled job from the job list or to show the job as paused or put on hold.

Referring to FIG. 5, a first case 60 shows how the queue manager handles jobs currently residing in the device queue 28. The queue manager 15 sends out a cancel request 62 that identifies the selected job and the selected function (cancel, pause, hold) to the device queue 28. If the selected job has not yet been output, the peripheral device deletes or holds the selected job in the device queue 28 according to the request 62 and then sends a confirmation 64 back to the queue manager 15.

Case 66 shows the cancel or pause operation for a job that currently resides in the server queue 24. The queue manager 15 identifies a job on the user interface that has been selected by the user for cancellation or pausing. The queue manager 15 identifies the server queue 24 as storing the selected queue and accordingly sends a cancel request 63 to the server queue 24. The network server deletes or pauses the selected job in server queue 24 according to the request 63 and sends out a confirmation 65 back to the queue manager 15. The queue manager 15 then updates the job list on the user interface to reflect the cancelled or paused job.

When a user is selecting jobs from the user interface display, some of the jobs may have already moved from the server queue 24 to the device queue 28. The queue manager automatically adjusts how the cancel or pause requests are sent according to where the job is currently located on the server queue 24 or device queue 28. Case 68 deals with the situation when a job 71 was in the server queue 24 when the user selected the job to be canceled or paused. However, the job 71 moved to the device queue 28 before the cancel request 70 was received by the server queue 24.

The queue manager 15 initially sends the cancel or pause request 70 to the server queue 24, since that was the last identified location for the selected job 71. Since the selected job 71 no longer resides on the server queue 24, the server queue 24 sends back a notice 72 to the queue manager 15 indicating that the cancel or pause request 70 failed.

The queue manager 15 upon receiving the fail notice 72 resends a cancel or pause request 74 to the device queue 28. The device queue 28 now stores the selected job 71. Therefore, the peripheral device sends a confirmation 76 back to the queue manager 15 after it cancels or pauses job 71. The queue manager 15 then updates the job list displayed on the user interface to reflect the cancelled or paused job 71.

Case 78 shows an alternative way to implement the condition where a job 81 moves from the server queue to the device queue 28 after the job 81 is selected. Instead of first sending a cancel request to the server queue 24, the queue manager 15 sends a cancel or pause request 80 to both the server queue 24 and the device queue 28 at the same time. This eliminates the queue manager 15 from having to identify which queue currently stores the job 81. As long as a cancel or pause confirmation 82 is received back from one of the queues 24 or 28, the queue manager 15 knows the cancel or pause request has been successfully completed.

FIG. 6 shows how the queue manager changes the priority of jobs. For example, a user may want to have one job in either the server queue or device queue be the next in line for printing from a peripheral device. If the job is currently fifth on the list of jobs to be output, the user can go to the user interface 16 (FIG. 1) and move the job up to the top of the list. The selected job will then be the next job to be output from the peripheral device. A user can either promote job priority or demote job priority by moving the job either up or down in the job list in field 30 (FIG. 2).

The jobs in the queues are displayed from the user interface in block 90. The queue manager 15 identifies a user changing the priority of a job in block 92. The queue manager in block 94 sends a request to one or both of the server queue and device queue that contains the priority operation selected by the user. After a confirmation from one of the queues is received in block 96, the queue manager in block 98 updates the user interface to reflect the job list with the selected job in the new promoted or demoted position.

FIG. 7 shows how the queue manager handles priority requests. In case 100, a user requests a promotion or demotion of a job within the list of jobs in the same queue. For example, the user may move the lowest priority job DOC6 in FIG. 2 only above the two other jobs DOC4 and DOC5 on the server queue 22. In another example, DOC3 in the MFP queue is moved up in priority above DOC2 in the same MFP queue.

Referring back to FIG. 7, the queue manager 15 sends a priority request 102 to the queue where the job priorities have to be changed. The queue 26 or 28 makes the priority change contained in the priority request 102 and sends a confirmation 104 back to the queue manager 104.

The job may have moved from one queue to another after the user selects a new priority for the job. Further, the job may be moved from an existing priority in the server

queue 24 to a selected priority in the device queue 28. This is referred to as moving a job across queue boundaries. Case 106 explains how the queue manager 15 operates when a job promotion goes across the queue boundaries. For example, job DOC5 in the server queue in FIG. 2 is moved above DOC3 in the MFP queue.

Referring to FIG. 1, the queue manager 15 sends request 108 to server queue 24 to promote job DOC5 to the top of the server queue 24. The job DOC5 is promoted to the top of the server queue 24 and is the next job that is sent to the device queue 28. The server queue 24 then sends a confirmation 110 to the queue manager 15.

The queue manager 15 sends a request 112 to device queue 28 to hold all jobs below the identified priority for job DOC5 and sends a request 114 to promote job DOC5. The device queue 28 holds all jobs having a lower priority below job DOC5 and then promotes job DOC5 to the requested priority when received from the server queue 24. After a confirmation 116 is sent by the device queue 28 that DOC5 has been promoted, the queue manager 15 sends a request 118 to release the jobs on hold.

In an alternative embodiment a slot is created for job DOC5 on the device queue 28. The job DOC5 is then moved to the slot. No jobs below the slot reserved for DOC5 can be output from the device queue 28 until the DOC5 job is output.

The device queue 28 may be full when the user moves job DOC5 ahead of one or more jobs in the device queue 28. If the device queue 28 is currently full, the server queue 24 may not be able to immediately send the selected job DOC5 to the device queue 28. In this situation, the device queue 28 may have to output some number of jobs before receiving the promoted job DOC5.

In one example, the user may have requested promotion of job DOC5 above DOC2 (FIG.2). However, it may be necessary to output jobs DOC1 and DOC2 before there is

enough space to receive job DOC5. The device queue 28 will not promote job DOC5 above DOC2 but only above DOC3 since DOC1 and DOC2 have to be output before job DOC5 can be moved to the device queue 28.

Case 122 deals with demoting a job across the queue boundaries. The user selects a job to be demoted in the device queue 28. The queue manager sends a request 124 to the device queue 28 to hold the selected job until all of the jobs with higher priority have been output. The device queue 28 sends back a confirmation 126. The demoted job may be shown moved on the user interface 16 but may not be physically moved in the device queue 28. For example, the device queue 28 may hold the selected job until the higher priority jobs have been output. Then the selected job is released for outputting.

The commands sent between the queue manager and the queues in one embodiment are sent using proprietary or standardized network protocols such as Simple Network Management Protocol (SNMP)/Job Management Information Base (MIB) or Internet Printing Protocol (IPP). Extensions to these network protocols may be added to provide some of the functions described above. Alternatively, a custom network protocol is used to perform these operations.

The system described above can use dedicated processor systems, micro controllers, programmable logic devices, or microprocessors that perform some or all of the queue operations. Some of the operations described above may be implemented in software and other operations may be implemented in hardware.

For the sake of convenience, the operations are described as various interconnected functional blocks or distinct software modules. This is not necessary, however, and there may be cases where these functional blocks or modules are equivalently aggregated into a single logic device, program or operation with unclear boundaries. In any event, the

functional blocks and software modules or described features can be implemented by themselves, or in combination with other operations in either hardware or software.

Having described and illustrated the principles of the invention in a preferred embodiment thereof, it should be apparent that the invention may be modified in arrangement and detail without departing from such principles. Claim is made to all modifications and variation coming within the spirit and scope of the following claims.